Research report 110

BLACKBOARD SYSTEMS

Iain D Craig

(RR110)

Abstract

The blackboard architecture is becoming an increasingly popular basis for the construction of problem-solving systems which operate in domains requiring qualitatively different kinds of knowledge to be applied in order to arrive at a solution to a problem. This paper presents the metaphor on which blackboard systems are based. The metaphor is then given an interpretation which constitutes the blackboard architecture. The structure of the blackboard database and its contents are considered from an evolutionary point of view. Finally, the paper considers some of the alternative approaches to the control of problem-solving in blackboard systems.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

November 1987

BLACKBOARD SYSTEMS

Iain D. Craig
Department of Computer Science
University of Warwick
Coventry CV4 7AL, UK

ABSTRACT

The blackboard architecture is becoming an increasingly popular basis for the construction of problem-solving systems which operate in domains requiring qualitatively different kinds of knowledge to be applied in order to arrive at a solution to a problem.This paper presents the metaphor on which blackboard systems are based. The metaphor is then given an interpretation which constitutes the blackboard architecture. The structure of the blackboard database and its contents are considered from an evolutionary point of view. Finally, the paper considers some of the alternative approaches to the control of problem-solving in blackboard systems.

1.INTRODUCTION

The blackboard architecture is becoming increasingly popular as a method for constructing systems to solve complex problems where different kinds of knowledge and expertise are needed. The process by which blackboard systems construct solutions is incremental and is based on the progressive application of a variety of knowledge to solution elements at varying levels of absraction.

The architecture is a comparatively informal construct and has evolved over the years since HEARSAY-II (Erman, 1975, 1980), the first blackboard system, was constructed. The informality of the construct means that there are certain points which cause disagreement amongst researchers: the major points of the architecture are, however, universally accepted. One of the reasons why the architecture remains informal is that it has been applied to a number of very different problems and each problem involves a slight re-interpretation of it.

Blackboard systems have been constructed to solve problems in speech understanding (HEARSAY-II), sonar interpretation (HASP/SIAP - Feigenbaum, 1978, Nii, 1982, 1986a,

1986b), errand planning (OPM - Hayes-Roth, 1979) and protein structure analysis (CRYSALIS - Terry, 1983; PROTEAN - Hayes-Roth, 1984, 1985b). The diversity of problems tackled inevitably leads to different perspectives on the central features of the architecture.

A final reason for informality is that the modern blackboard architecture is a rationalisation of the HEARSAY-II structure.

Rather than attempt a universally acceptable definition of the architecture, this paper is concerned with tracing the major components of the modern interpretation of the architecture from HEARSAY-II to BB1 (Hayes-Roth, 1984, 1985a, 1985b, 1986) and GBB (Johnson, 1987a, 1987b). The paper is structured as follows.

In the next section, the blackboard metaphor is introduced. The metaphor is one of co-operative group problem-solving. Following the presentation of the metaphor, the basic components of the architecture are described. This is intended to constitute an uncontroversial statement of those aspects of the architecture accepted as being necessary before a system may be given the title 'blackboard system'.

In section three, the structure of blackboard systems will be considered. The form of this examination is a development of the structure of the blackboard database and its contents.

Section four deals with the problem of control. This is a crucial aspect because blackboard systems solve problems involving immense search spaces. The control of problem-solving activity has been an active research area for some years and the tendency to explicitly represent control can be seen in many blackboard systems.

Section five acts by way of a conclusion. It summarises the development of the architecture.

Unfortunately, in a paper of this size, it is not possible to examine in great detail the architecture and the systems mentioned above. Interested readers should consult Nii (1986a, 1986b) for a more detailed treatment and a comprehensive bibliography.

1.1 Acknowledgements

I would like to thank Peter Harper and David Wilson for many interesting discussions on the architecture and on the implementations described below. Many thanks are due to Liz Woolley

for drawing the figures.

2. THE ARCHITECTURE

2.1 Introduction

The blackboard architecture is based on a metaphor of group problem-solving first introduced by Newell and later re-interpreted by Simon. Newell took the metaphor and developed it into the production rule architecture (Newell, 1973; Davis, 1977; Waterman, 1978). It was Simon who suggested the metaphor to the HEARSAY team: it is from this suggestion that blackboard systems developed. In this section, the metaphor will be introduced and the major points of the architecture outlined.

2.2 The Metaphor

The blackboard metaphor is due to Newell: he later expanded upon the metaphor (Newell, 1969):

> "Metaphorically, we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it and to judge when he has something worthwhile to add to it"

Newell wanted to derive an architecture which did not suffer from the constraints of the then current fixed procedure-calling sequence and was aiming for something with demon-like properties.

If the metaphor is considered in detail, it can be seen that it suggests a group of agents, each an expert in some field. The group is assembled to solve a complex problem using a blackboard and chalk. The agents are allowed to write or draw on the blackboard to record their contributions to the solution in a way which is accessible to other group members. An agent is only permitted to respond to items on the blackboard which fall within their individual areas of expertise.

In the light of the implemented blackboard systems, it is necessary to expand the metaphor slightly. The agents are now allowed to draw arrows between items on the blackboard so that they can be collected together to form 'islands' in the emerging solution. The agents can focus

on any particular island if it seems interesting or promising. In addition, the agents must be required to be deaf and dumb: that is, all and any communication between agents must be in terms of things written or drawn on the blackboard.

The metaphor contains all the elements important to the blackboard architecture. There is a central blackboard on which all results of problem-solving activities by individual agents are recorded. Anything written on the blackboard can, potentially, be of use to any agent in the group. It is not necessary for an agent to write something special that is aimed at interesting another agent: instead, agents place things on the blackboard when they consider they have a contribution to make. The ways in which items placed on the blackboard are derived does not matter: it is not possible for one agent to examine the details of another's problem-solving processes. Since the entire group is involved in constructing a solution, and since each agent's internal thought processes are not inspectable by others, if one agent is removed from the group for some reason - say illness or another appointment - only the quality of the solution will differ. These last two points can be summarised by saying that agents are modular, independent processes.

2.3 The Blackboard Architecture

The blackboard architecture is based on the salient features of the HEARSAY-II system. Here, the major features of the architecture will breifly be examined.

Blackboard systems are composed of three main components:

- A globally accessible database called the *blackboard*. The blackboard is structured as a linear hierarchy of *abstraction levels*. The blackboard contains the results of applying problem-solving knowledge.
- A set of *Knowledge Sources*. Knowledge Sources represent the problem-solving knowledge contained in a blackboard system. They respond to changes to the state of the blackboard database by altering its contents.
- A control component called the *scheduler*. The scheduler is charged with controlling the

problem-solving behaviour of the system. It does this by monitoring the current state of the blackboard and selecting one or more Knowledge Sources to apply to it.

Each of these components requires a little explanation.

The blackboard database (or simply the *blackboard*) holds all intermediate results of problem-solving. It is used by all Knowledge Sources as their only means of communication: they may *only* communicate by modifying items already on the blackboard or by adding new items. The blackboard is organised as a linear abstraction hierarchy: this allows the problem under attack to be analysed at different levels of description. The abstraction levels (or *levels*) on the blackboard constitute an outline plan for solving the problem.

The items placed on the blackboard are called *entries* (they are given other names by various system builders - for example, in HEARSAY-II, they were called *hypotheses*). Entries represent elements of the solution being developed and are usually complex structures. The most common representation for entries is attribute-value pairs: that is, each entry is composed of a set of attributes which are assigned values by Knowledge Sources. The attributes appearing in entries will vary with problem and with abstraction level. Nii (1986a) recommends that each abstraction level be assigned its own representational vocabulary: that is, the entries at each level of abstraction have different sets of attributes. HEARSAY-II, on the other hand, used a uniform vocabulary: i.e., all entries had the same attributes. The idea that each abstraction level has its own vocabulary is attractive because it not only defines what can be represented on each level, but it also serves to constrain Knowledge Sources; in addition, the use of distinct attribute sets can be of considerable value when testing a system.

Entries are often linked together to form *solution islands*. An entry may be linked to other entries on the same, on higher, or on lower abstraction levels. It is frequently the case that the solution to the problem being solved does not consist simply of one entry or of one assertion, but of a solution island. Links between entries forming solution islands represent structural relationships on the blackboard. For example, a link from an entry to an entry on a lower level might represent the fact that the information held in the more abstract entry is justified by the information in the lower one. One entry may have links to a number of others. In HEARSAY-

II, the links were organised as an AND/OR graph; in HASP/SIAP, they were organised as an AND graph. The purpose of the link structure is to assemble individual entries into collections which constitute potential solutions.

Knowledge Sources represent the problem-solving knowledge in blackboard systems. They are structured as condition-action pairs.

The condition-part monitors the blackboard for changes such as the addition of a new entry or the modification of an already present one: such a change is often referred to as a *blackboard event*. Some systems, such as HASP/SIAP, allowed other events to be monitored by Knowledge Source conditions: an example is response of some Knowledge Sources to the timing pulses generated by the computer's real-time clock. Condition-parts can also monitor the blackboard state: this amounts to testing islands or entries to see if they satisfy some predicate.

The action-part of a Knowledge Source makes changes to the blackboard. It can add or modify entries or solution islands. It is rare for an action-part to delete entries from the blackboard (but see Craig, 1987, for a promising approach). The action-part (henceforth: *action*) is executed only after the condition-part has been satisfied. When the condition-part of a Knowledge Source has been satisfied, the Knowledge Source is usually referred to as having been *triggered*. A triggered Knowledge Source is eligible for execution: it may be considered by the scheduler for execution of its action.

The final component specified by the architecture is the *scheduler*. The scheduler is responsible for implementing one or more problem-solving strategies which guide the system's problem-solving activity. It acts upon triggered Knowledge Sources and uses problem-specific parameters to determine which Knowledge Source action to execute. There is no prescription that only one action should be executed at any one time: in a parallel environment, a collection of Knowledge Sources may be executed concurrently. The scheduler examines the blackboard state and those Knowledge Sources which have been triggered and makes its selection on the basis of the prescriptions of the strategy currently in force. It is possible for blackboard schedulers to interrupt one strategy in order to adopt another if the blackboard state requires it: this is one of the more powerful aspects of the architecture.

The term *opportunism* is frequently associated with blackboard systems. Opportunism can be

defined as a control strategy which makes the best use of the current solution state. For example, if the current strategy is to develop a solution from high to low levels of abstraction and there is a solution island which seems promising, but which does not occupy a region of the blackboard currently in the focus of the top-down strategy, the scheduler may select Knowledge Sources to act on the promising island, ignoring the prescriptions of the current strategy. Blackboard systems are particularly suited for this kind of situation-determined behaviour, but the rôle of opportunism is somewhat controversial. Nii (1986a) requires it to be part of the architecture definition; Hayes-Roth (1983) states only that it is a characteristic. In (Craig, 1987b), it has been argued that opportunism is *not* part of the basic definition because it is a class of strategy.

3. ARCHITECTURAL DEVELOPMENTS

3.1 Introduction

The various blackboard system implementations have not all exactly followed the structure of HEARSAY-II. The basic architectural principles are that there should be a hierarchical, global database, a collection of Knowledge Sources and a scheduler. In this section, some of the structural developments of the architecture are traced.

3.2 Blackboard Organisation

An early (1975) version of HEARSAY-II had a single blackboard divided into seven abstraction levels. HEARSAY-II's task was to process speech in near real-time. The organisation of the blackboard is shown in figure 1. It should be noted that the system had an extra level not included in the figure, the DATABASE INTERFACE level, merely interfaced the system to a text database: the level appeared at the top of the abstraction hierarchy, immediately above the PHRASAL level.

The Hearsay-II blackboard was arranged as a single, linear abstraction hierarchy. At the bottom of the hierarachy, digitised speech signals entered the system and were represented on the PARAMETER level. Features of the signal were progressively abstracted until phrases could be hypothesised. The phrasal hypotheses were represented on the PHRASAL level and were

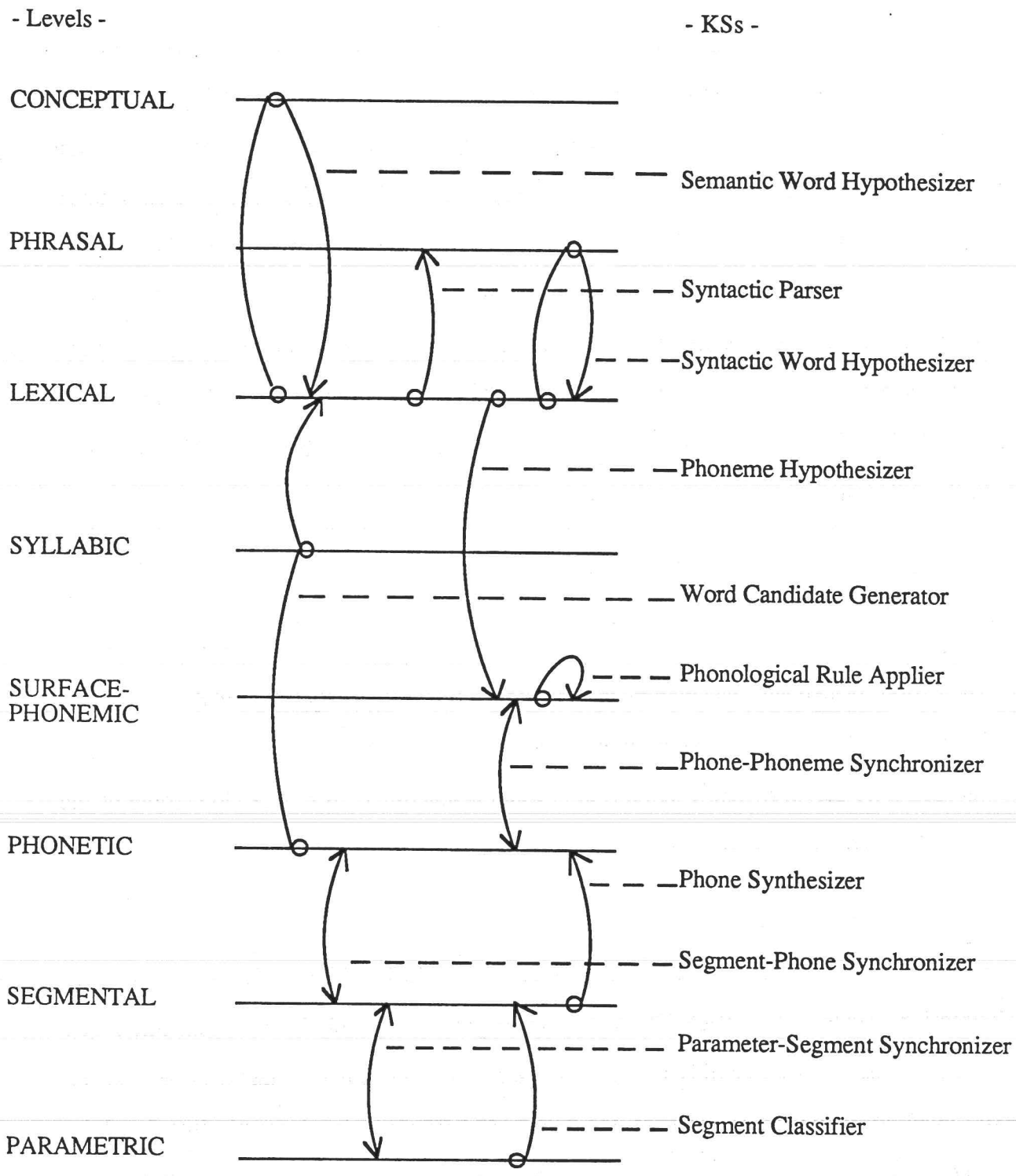

Figure 1

HEARSAY-II Knowledge Sources (1975)

composed of sequences of words which had been hypothesised on the WORD level. The system operated in a basically bottom-up fashion: hypotheses on lower abstraction levels were aggregated to form higher level ones.

As can be seen from figure 1, the blackboard was organised according to the components of a phrase. The ultimate goal of the system was to find syntactically correct, meaningful phrases in the speech signal so that database items could be retrieved. The PHRASAL level of the blackboard, therefore, represents the most abstract representation of the speech signal. A similar analysis underpins the HASP/SIAP and PROTEAN blackboard structures: the HASP/SIAP blackboard is shown in figure 2.

Entries on the HEARSAY-II blackboard were represented using a uniform attribute set. The attributes were used to record features of the speech signal at various levels of abstraction and also contained links to other entries. Since time is an important aspect of speech processing, there was a set of attributes devoted to its representation. Associated with each entry, there was a complex attribute which recorded the confidence the system had of the hypothesis. In addition, entries had attributes to record information about how many resources (processor time and memory) had been expended on them, as well as holding recommendations as to future processing: these attributes were used by the scheduler.

An entry could only reside at one level of abstraction. It was not possible, for example, for an entry to represent something mid-way between a word and a collection of syllables (this is a necessary property of entries in all blackboard systems).

The HEARSAY-II blackboard organisation stands as a model for all current implementations. However, there are times when it is not possible to generate a single abstraction hierarchy: this was the case with the OPM (Hayes-Roth, 1979) and CRYSALIS (Terry, 1983) systems. Although CRYSALIS was the first to introduce extra structural divisions on the blackboard, the OPM task is easier to explain.

OPM was designed to model the behaviour of human subjects planning a sequence of errands in a hypothetical town. The subjects were given a list of errands to run and a time within which they should complete them: the time allocated was insufficient for all errands to be run. Verbal protocols from the subjects were used as the basis for building OPM. The basic method which

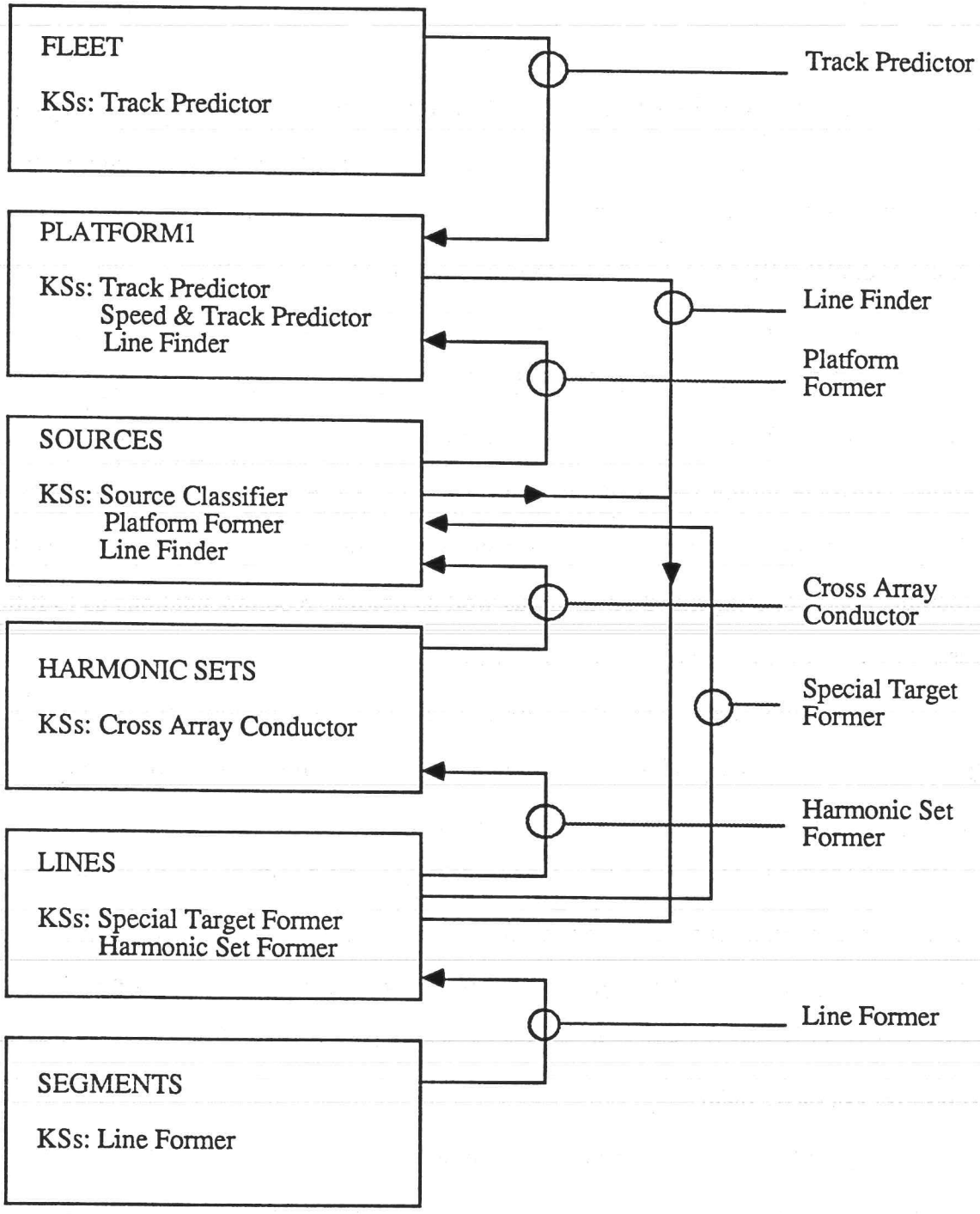

Figure 2

HASP/SIAP Blackboard

was discovered for errand planning was opportunistic: that is, subjects would select errands on the basis of the physical proximity and not necessarily in a top-down fashion.

From the analysis, additional structural divisions of the blackboard were introduced into OPM. These extra divisions, called *planes* or *panels*, are workspaces within which to solve problems in different subject domains or which concern different topics. OPM contained five planes: the PLAN, PLAN-ABSTRACTION, KNOWLEDGE BASE, EXECUTIVE and META-PLAN planes. As can be seen from the names, each plane contained entries which dealt with qualitatively different kinds of knowledge, although all but one are concerned with planning: this exception is the EXECUTIVE plane which is charged with controlling the execution of the system and with the implementation of problem-solving strategies.

Each plane had its own, local abstraction hierarchy. A plane can be viewed as a complete blackboard in its own right: under this interpretation, OPM (and CRYSALIS) was really a collection of blackboard systems. Communication of information between planes was performed by Knowledge Sources which triggered on one plane and alter the state of another. Planes are useful when a problem cannot be decomposed into a single, uniform abstraction hierarchy. In the case of OPM, each plane is justified on the basis of the knowledge deployed by the human subjects when planning their errands.

The Hearsay-III system (Balzer, 1980; Erman, 1981) added extra partitions to the blackboard. Hearsay-III was designed to be a domain-independent shell with which to construct blackboard systems (as is AGE (Nii, 1979, 1980)). Hearsay-III was constructed on top of a powerful relational database system, called AP3 (Goldman, 1978), which provided a number of features including strong-typing and constraints. The Hearsay-III blackboard was divided into a collection of *contexts* which provided non-overlapping blackboard regions which spanned abstraction levels. Contexts were used to ensure that different lines of reasoning be kept distinct: this facilitated control of inference because non-overlapping solutions could be independently focussed upon - there was no risk that an overlap of partial solutions could cause Knowledge Sources to diverge from the current focus.

The context mechanism added an extra structural level to the blackboard, but, unlike the abstraction hierarchy and divisions into planes, contexts were generated dynamically as a result

of Knowledge Source execution. Contexts were also hierarchically organised, so sub-contexts could inherit information from super-contexts: this allowed regions of the blackboard to be sub-divided as much as necessary.

The GBB system (Johnson, 1987a, 1987b) is another domain-independent tool for the construction of blackboard systems. In GBB, the central blackboard database is composed of a set of blackboards. Each blackboard is a hierarchical structure composed of atomic components called *spaces*. A blackboard can also be composed of other blackboards. A space is a region of highly structured storage and is organised by dimensions. Dimensions are used to locate objects using terms natural to the application domain. GBB supports two types of dimension: enumerated and ordered. Enumerated dimensions are similar to enumeration types in Pascal or ADA; ordered dimensions allow objects to be retrieved using a numeric range. In the GBB implementation of BB1, for example, the enumeration *{domain, control}* is used to index the Knowledge Sources in the system.

The objects stored on the GBB blackboard are called units. They are structured objects with an attribute-value format. When a unit is created, it has five pre-defined slots: slots, links, path-indexes, paths and dimensional indexes. Links are special-purpose slots which contain links between units. Links perform a rôle directly analogous to links in HEARSAY-II. Paths and path-indexes are used to determine the spaces in which a unit should be stored. Dimensional indexes are used to determine where on its space or spaces, a unit should be located.

The aim of GBB is to make the blackboard as fast as possible and to provide an environment in which users can specify system components in as natural a fashion as possible. The efficiency considerations which underpin GBB are important in many domains and have frequently been ignored.

4. CONTROL

4.1 Introduction

Control is a crucial aspect in building blackboard systems. The blackboard organisation

provides support for control in the form of the abstraction hierarchy, but it is the scheduler which enforces control strategies. The scheduler makes control decisions on the basis of the state of the blackboard and the applicable knowledge which is available at any point in time. Blackboard schedulers can implement multiple strategies and are able to switch between strategies depending on the state of the emerging solution.

It is common for the scheduler to maintain a control database: this records Knowledge Source activations in some form or another. The scheduler searches its control database to find Knowledge Sources which are applicable to the current state. Knowledge Source activations are created whenever a Knowledge Source is triggered: they record a variety of information about what caused the Knowledge Source to trigger (details of the triggering event, that is), where the Knowledge Source triggered as well as other information useful to the scheduler. The control database records Knowledge Source activations from previous cycles: this contrasts with other archtectures which only record information about the current interpreter cycle. Not all blackboard systems employ this form of control: HASP/SIAP and AGE (Nii, 1979), for example, adopted a slightly different régime.

4.2 Two Approaches to Scheduling

There are two basic approaches to scheduling blackboard system actions: they are called knowledge- and event-scheduling. The scheduler applies some strategy to determine which Knowledge Source actions should be executed next. The distinction between knowledge- and event-scheduling is the way in which they come to a decision about which actions to execute. In knowledge-scheduling systems, the problem is of finding the most appropriate Knowledge Source to execute. In event-scheduling systems, the problem is one of selecting the event to which knowledge is to be applied. That is, whenever there is an event on the blackboard (or from a timer, say), some Knowledge Sources are triggered. Over time, as the number of candidate events increases (because they have not been selected as the focus of attention), there is a choice as to which event to select next.

The primary constraint on scheduling, of whatever form, is that any Knowledge Sources which are considered must have been triggered, and must, therefore, be present in the control

database.

Knowledge-scheduling, because it amounts to operator selection, implies some form of planning process. Event-scheduling, because there is not, generally, a procedure to determine what the candidate events will be, cannot rely upon mechanisms of that sort.

However a system makes its scheduling decisions, it is necessary for it to employ problem-solving strategies. This is so because strategies prescribe parameter values which can form the basis of control: it is also necessary for some system component to have an overview of the current solution state so that it can direct the actions of the system.

4.3 Example Systems

HEARSAY-II employed what has become known as an intelligent scheduler: this is a scheduler coded in the implementation language which contains procedural representations of the problem-solving strategies to be used by the system. HEARSAY-II operated two basic strategies: an initial bottom-up aggregation of hypotheses as far as the WORD level, followed by an opportunistic phase during which higher level hypotheses were generated. The opportunistic phase employed, to a rough approximation, a strategy which selected Knowledge Sources to execute on the basis of the confidence level of the entries on which they had triggered: the strategy worked by trying to find the best region of the blackboard to expand next. In addition, HEARSAY-II applied sub-strategies of various kinds: for example, best-first search and means-end analysis.

The HEARSAY-II scheduler operated on a queue, called an *agenda*, of Knowledge Source Activation Records (*KSAR*s). Each KSAR represented a triggered Knowledge Source and contained additional information to allow the scheduler to make decisions. The best-first and means-end analysis searches were performed on the agenda by the scheduler and by searching the blackboard to obtain information about the state of the emerging solution. The need to search the blackboard to make scheduling decisions was an major problem in the system. A detailed account of one of the schedulers used in HEARSAY-II can be found in Hayes-Roth (1977).

The HASP/SIAP system employed a totally different approach. It was designed for

continuous operation and monitored the sonar signals emitted by the moving parts of vessels in an ocean region. The input data came into the system in small chunks and was posted on the lowest level of the blackboard. From this data, together with intelligence reports, the system had to identify the vessels in the area under observation. The problem tackled by the system was essentially event-based, so event-scheduling was used. The incoming evetns supported an expectation-based strategy, whereas intelligence reports were used as part of a model-based one.

In HASP/SIAP there were three main event classes. The classes were: *blackboard* events, *expected* events and *clock* events. *Blackboard* events were events generated by the application of Knowledge Sources to the emerging solution. *Expected* events were events which one or more Knowledge Sources expected to occur at some later time. Finally, *clock* events were driven by the system clock: they allowed for synchronisation with the external world, amongst other things.

When an event occurred, one or more Knowledge Sources would trigger and would be placed in a control database. HASP/SIAP maintained a control database for each event type and they were searched in FIFO order.

The control structure in HASP/SIAP was composed of a hierarchy of control Knowledge Sources, the highest of which implemented the problem-solving strategy in force. At the bottom of the hierarchy, there were rules to associate events with Knowledge Sources: when an event occurred, one or more rules would fire and appropriate Knowledge Sources would have their precondition tested against the event. In the middle of the hierarchy, there were event-list managers: these Knowledge Sources maintained the control databases.

The Strategy Knowledge Source was responsible for implementing the strategies used by HASP/SIAP. There were two major strategies in the system: *model-* and *expectation-driven.* In *model-driven* processing, an abstract model of a situation is used to drive problem-solving. A model will contain features which are expected to occur in a situation and Knowledge Sources to detect these features can be scheduled on the basis of what is required to form the model on the blackboard. Models also enable a system to determine what is important by classifying information. In *expectation-based* processing, events are predicted to occur on the basis of

what has already happened: for example, if it is known that an event of type E_1 has occured and events of this type are known to be usually followed by events of type E_2, Knowledge Sources which respond to E_2 events can be scheduled based on that expectation.

The scheduling techniques employed by HASP/SIAP involve the explicit use of strategies. The mechanisms for applying them is not, though, fully integrated with the architecture. An integration of control with the architecture is to be found in the Blackboard Control Architecture of Hayes-Roth (1985a). It is not possible to give a detailed account of the Control Architecture within the space available here: the Architecture is complex and fairly difficult to understand, so the reader is advised to consult Hayes-Roth (1985a) for a more complete account.

The Control Architecture uses two fundamental divisions of the blackboard: the *domain* and *control* blackboards (each of which corresponds to a plane). The domain blackboard is used to represent reasoning about the problem under attack; the control blackboard is used to represent control information. The basic unit of control is the KSAR: KSARs are created whenever a Knowledge Source is triggered by a blackboard event. When created, a KSAR is placed in an agenda. The agenda is divided into two sub-agendae: one for triggered Knowledge Sources and one for Knowledge Sources which are eligible for execution. A Knowledge Source is eligible for execution when its precondition has been satisfied as well as its trigger. The scheduler inspects the eligible KSARs and selects one for execution.

The scheduler in the Control Architecture is the control blackboard. That is, control is explicitly represented as another problem for the system to solve. In this architecture, control is viewed as a planning process and is, therefore, a form of knowledge-scheduling. The control blackboard contains a plan for solving the problem under attack and contains an abstraction hierarchy. Reasoning on the control blackboard is about the selection of the next KSAR to execute. The highest level contains entries which describe the problem the system is to solve. The lowest levels, the TO-DO-SET and CHOSEN-ACTION levels represent control information at the most detailed level. The TO-DO-SET level contains the system agenda: each entry on this level represents an encapsulation of the blackboard state on each cycle. The CHOSEN-ACTION level records the KSAR which was selected for execution. Above these levels, the control problem is represented in terms of strategy and focus decisions: it can be augmented by

heuristic and tactical decisions - these are used to finesse the higher level planning decisions (the TACTIC level is used in a model of HEARSAY-II's control structure; the HEURISTIC level is used in the PROTEAN system). The control blackboard structure for a version of the OPM system using the control architecture is shown in figure 3: the reader should note the organisation of abstraction levels for the CONTROL components (lower box).

The Control Architecture represents strategic decisions as entries on the control blackboard's STRATEGY level. A strategy decision is usually implemented in terms of a series of focus-of-attention decisions represented on the FOCUS level. A strategy determines which scheduling parameters FOCUS level Knowledge Sources should rate most highly. For example, a strategy decision might specify that successively lower domain blackboard abstraction levels should be focused upon (thus implementing a strictly top-down strategy beginning at the highest level and ending at the lowest). Such a decision generates a set of FOCUS decisions - one for each domain abstraction level - in a strict order. The FOCUS level Knowledge Sources then act on each FOCUS decision and will make decisions about the relative worth of the parameters instantiated in KSARs created by Knowledge Source triggering on the currently focussed abstraction level. For each focus of attention, only those KSARs specified by the current FOCUS decision are examined and other parameters are evaluated.

FOCUS level decisions are used to order the contents of the agenda of eligible KSARs. The highest rated KSAR is then selected for execution and its KSAR is posted on the CHOSEN-ACTION level to indicate that it has been executed.

The control architecture is able to represent the scheduling process in terms of explicit decisions taken by control Knowledge Sources: these Knowledge Sources post control decisions on the control blackboard and also monitor the execution of domain Knowledge Sources. The architecture, because it provides a domain-independent framework within which to represent problem-solving can provide a variety of different control strategies: the user defines control Knowledge Sources which post decisions to the appropriate abstraction level of the control blackboard and which respond to various events and states in the system. The Control Architecture can easily provide top-down and bottom-up, as well as other, generic control strategies; it can also be used to implement domain-specific strategies. Because all

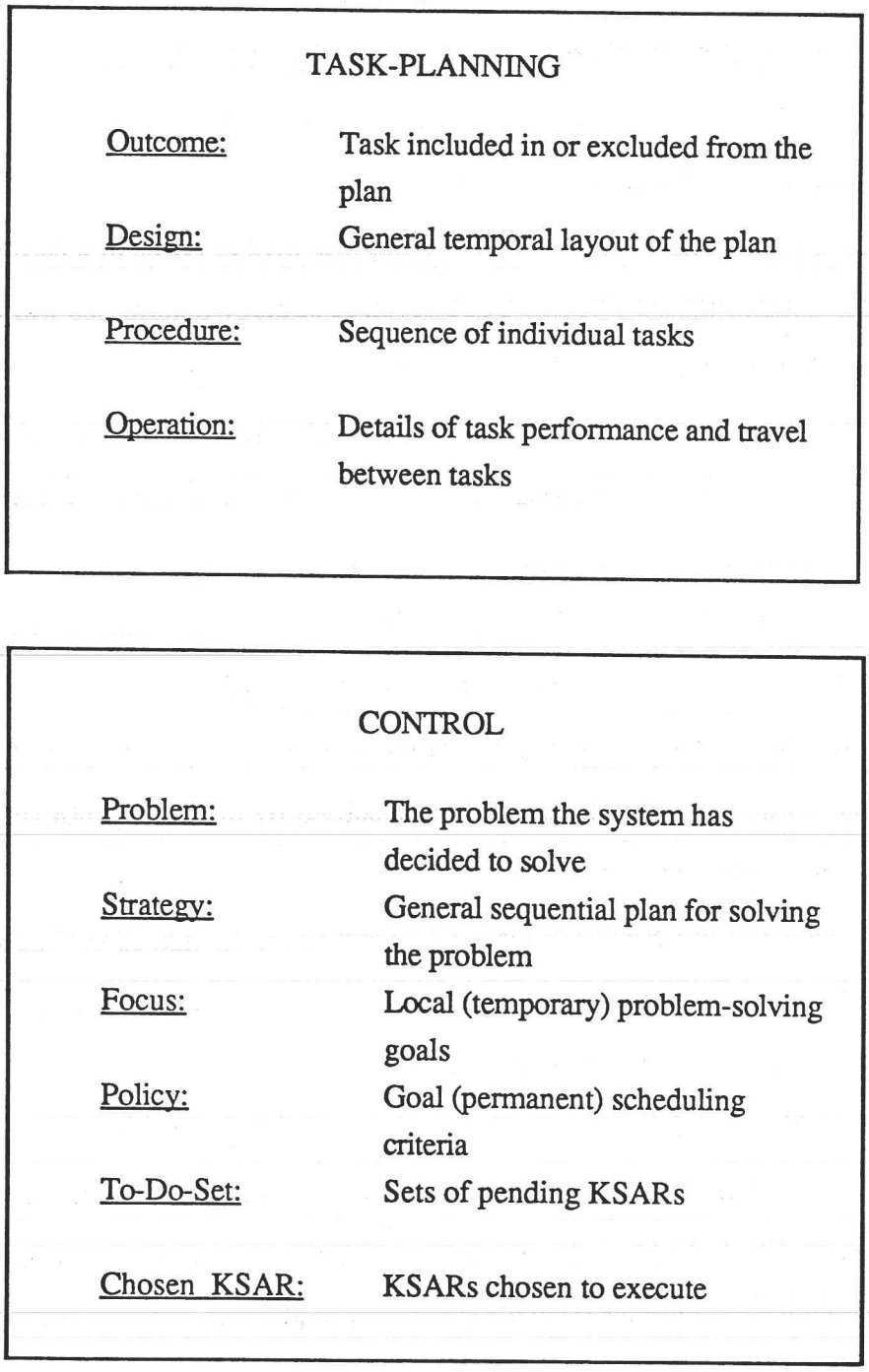

Figure 3

Abstraction levels for the OPM blackboard
(level names are underlined)

control information is explicitly represented in the Architecture, opportunistic strategies can be used: one abstract example is the strategy which examines the KSARs in the eligible agenda and selects the one with the highest value on one of its attributes. It is also possible, very easily, to change between strategies and foci: this is because control Knowlegde Sources monitor explicit representations of scheduling decisions and parameters.

More recently (Hayes-Roth, 1986), the Control Architecture has been augmented with a declarative knowledge base which contains representations of the goals an application system has to satisfy. The knowledge base allows explicit goal-directed reasoning to be performed in systems. This extension is reminiscent of the Goal-and-Data blackboard system developed by Lesser and Corkhill (Corkhill, 1982; Durfee, 1987) in which the blackboard is divided into two planes, one for goals, the other for data. When data arrives in the system, one or more goals is activated and used to drive a planning process. When a goal becomes active, Knowledge Sources which can be used to satisfy it are considered for execution: this may either involve the generation of sub-goals or may require that the system wait for the arrival of new data before the satisfaction process can be completed. It is, of course, possible for the system to suspend its focus on one goal in order to re-direct its attention to another

Both the Blackboard Control Architecture and the Goal-and-Data architecture contain explicit representations of the control process and both treat control as being fundamentally a planning activity. Both of these approaches can be used to constrain behaviour while still allowing a rich variety of potential control decisions. They can both be used either to control the amount of opportunistic behaviour exhibited. These systems contrast strongly with the event-based approach adopted in HASP/SIAP which requires control to provide appropriate responses to events occuring in real-time. Although there are similarities between the tasks performed by HASP/SIAP and the Distributed Vehicle Testbed system (Lesser, 1983), HASP/SIAP differs in that it is not known *a priori* that a particular course of events will actually occur: in other words, it is not known that a solution to the problem exists. As a conclusion to this section, it is suggested that event-scheduling is best used when it is not possible to plan a solution and that knowledge-scheduling be used whenever planning is possible.

5.CONCLUSIONS

This paper has surveyed the basic concepts of blackboard systems and has shown some aspects of their development in some of the more famous implementations.

In particular, the paper has concentrated on the development of the following aspects of the architecture. The blackboard database structure has been traced from the single, linear hierarchy employed in HEARSAY-II to the multiple-panel structure to be found in OPM. The structure of entries on the blackboard has also been considered. In HEARSAY-II, entries had one basic representational vocabulary which spanned the entire blackboard: in later systems, individual abstraction levels have tended to have their own vocabularies which are tailored to optimal representation of the objects which reside there.

Control, it has been suggested, is a crucial aspect of blackboard systems. Control structures in blackboard systems have developed from the monolithic, intelligent scheduler to the explicit representations of the Blackboard Control Architecture. The development of control can be seen to be a progressive integration of the blackboard architecture with the needs of producing solutions in an acceptable time: this has led to the systematic exploitation of problem-solving strategies and extensions to the architecture which allow effective use of the information stored on the blackboard and in the scheduler's database.

Currently, there is considerable work on the blackboard architecture. Much of this work is centred on its application to an increasing number of problem domains. Although there is still some disagreement on the theoretical level as to precisely what constitutes a blackboard system, it is becoming clear that the architecture is stable enough to become a viable alternative for the construction of application systems.

REFERENCES

(Balzer, 1980) Balzer, R., Erman, L.D., London, P. and Williams, C., Hearsay-III: A domain independent framework for expert systems, *Proceedings of the First National Conference on Artificial Intelligence*, pp. 108 - 110, 1980

(Corkhill, 1982) Corkhill, D.D., Lesser, V.R. and Hudlicka, E. Unifying data-directed and goal-directed control: An example and experiments, *Proceedings of the AAAI*, pp. 143 - 147, 1982

(Craig, 1987) Craig, I.D., Decentralised Control in a Blackboard System, Ph.D. Thesis, Department of Computing, University of Lancaster, 1987

(Craig, 1987b) Craig, I.D., The Blackboard Architecture: A Definition and its Implicatins, Research Report No. RR98, Department of Computer Science, University of Warwick, 1987

(Davis, 1977) Davis, R. and King, J., An Overview of Production Systems *in* Elcock, E.W. and Michie, D. (Eds.), *Machine Intelligence 8*, John Wiley, 1977

(Durfee, 1987) Durefee, E.H., Lesser, V.R. and Corkhill, D.D., Cooperation Through Communication in a Distributed Problem Solving Network, *in* (Huhns, 1987), pp. 29 - 58, 1987

(Erman, 1975) Erman, L.D. and Lesser, V.R. A multi-level organization for problem solving using many, diverse, cooperating sources of knowledge, *Proceedings of the Fourth International Joint International Conference on Artificial Intelligence*, Vol. 2, pp. 483 - 490, 1975

(Erman, 1980) Erman, L.D, Hayes-Roth, F., Lesser, V.R. and Reddy, D.R., The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty, *Computing Surveys*, Vol. 12, pp. 213 - 253, 1980

(Erman, 1981) Erman, L.D., London, P.E. and Fikas, S.F., The design and an example use of Hearsay-III, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 409 - 415, 1981

(Feigenbaum, 1978) Feigenbaum, E.A. and Nii, H.P., Rule-based understanding of signals *in* (Waterman, 1978)

(Goldman, 1978) Goldman, N., AP3 User's Guide, USC/ISI, Marina del Rey, CA, 1978

(Hayes-Roth, 1977) Hayes-Roth, F. and Lesser, V.R., Focus of Attention in the HEARSAY-II Speech Understanding System, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* pp. 1014 - 1029, 1977

(Hayes-Roth, 1979) Hayes-Roth, B. and Hayes-Roth, F., A Cognitive Model of Planning, *Cognitive Science*, Vol. 3, pp. 275 - 310, 1979

(Hayes-Roth, 1983) Hayes-Roth, B. The Blackboard Architecture: A general framework for problem solving? Report No. HPP-83-30, Computer Science Department, Stanford University, Palo Alto, CA, 1983

(Hayes-Roth, 1984) Hayes-Roth, B. BB1: An architecture for blackboard systems that control, explain and learn about their own behavior, Technical Report HPP-84-16, Computer Science Department, Stanford University, Palo Alto, CA, 1984

(Hayes-Roth, 1985a) Hayes-Roth, B. A blackboard architecture for control, *Artificial Intelligence Journal*, Vol. 26, pp. 251 - 321, 1985

(Hayes-Roth, 1985b) Hayes-Roth, B. and Hewett, M. Learning Control Heuristics in a Blackboard Environment, Technical Report HPP-85-2,Computer Science Department, Stanford University, Palo Alto, CA, 1985

(Hayes-Roth, 1986) Hayes-Roth, B., Garvey, A., Johnson, M.V. and Hewettt, M., A Layered Environment for Reasoning about Action, Knowledge Systems Laboratory, Report No. KSL 86-38, Computer Science Department, Stanford University, Palo Alto, CA, 1986

(Huhns, 1987) Huhns, M.N. (Ed.) Distributed Artificial Intelligence, Pitman, London, 1987

(Johnson, 1987a) Johnson, P.M., Corkhill, D.D. and Gallagher, K.Q., Integrating BB1-Style Control into the Generic Blackboard System, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1987

(Johnson, 1987b) Johnson, P.M., Gallagher, K.Q. and Corkhill, D.D., GBB Reference Manual, Technical Report 87-36, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1987

(Lesser, 1983) Lesser, V.R. and Corkhill, D.D., The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks, *Artificial Intelligence Magazine*, Vol. 4, No. 3, pp. 15 - 33, 1983

(Newell, 1969) Newell, A., Heuristic Programming: Ill-structured Problems in Progress in *Operations Research*, Vol. 3, 1969

(Newell, 1973) Newell, A., Production systems: models of control structures in Chase, W.G.,(Ed.) *Visual Information Processing*, Academic Press, New York, 1972

(Nii, 1979) Nii, H.P. and Aiello, N., AGE: A knowledge-based program for building knowledge-based programs, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pp. 645 - 655

(Nii, 1980) Nii, H.P., An Introduction to Knowledge Engineering, Blackboard Model and AGE, Report No. HPP-80-29, Computer Science Department, Stanford University, Palo Alto, CA, 1980

(Nii, 1982) Nii, H.P., Feigenbaum, E.A., Anton, J.J. and Rockmore, A.J., Signal-to-symbol transformation: HASP/SIAP case study, *Artificial Intelligence Magazine*, Vol. 3, pp. 23 - 35, 1982

(Nii, 1986a) Nii, H.P., The Blackboard Model of Problem Solving, *Artificial Intelligence Magazine*, Vol. 7, No. 3, pp. 38 - 53, 1986

(Nii, 1986b) Nii, H.P, Blackboard Systems Part Two: Blackboard Application Systems, *Artificial Intelligence Magazine*, Vol. 7, NO. 3, pp. 82 - 106, 1986

(Terry, 1983) Terry, A., The CRYSALIS Project: Hierarchical Control of Production Systems, Memo HPP-83-19, Computer Science Department, Stanford University, Palo Alto, CA, 1983

(Waterman, 1978) Waterman, D.A. and Hayes-Roth, F., Pattern-directed inference systems, Academic Press, New York, 1978